



Linux-Foundation

Exam Questions CKS

Certified Kubernetes Security Specialist (CKS) Exam

NEW QUESTION 1

Create a new NetworkPolicy named deny-all in the namespace testing which denies all traffic of type ingress and egress traffic

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

You can create a "default" isolation policy for a namespace by creating a NetworkPolicy that selects all pods but does not allow any ingress traffic to those pods.

```
--
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny-ingress
spec:
  podSelector: {}
  policyTypes:
  - Ingress
```

You can create a "default" egress isolation policy for a namespace by creating a NetworkPolicy that selects all pods but does not allow any egress traffic from those pods.

```
--
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-all-egress
spec:
  podSelector: {}
  egress:
  - {}
  policyTypes:
  - Egress
```

Default deny all ingress and all egress traffic You can create a "default" policy for a namespace which prevents all ingress AND egress traffic by creating the following NetworkPolicy in that namespace.

```
--
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny-all
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  - Egress
```

This ensures that even pods that aren't selected by any other NetworkPolicy will not be allowed ingress or egress traffic.

NEW QUESTION 2

Fix all issues via configuration and restart the affected components to ensure the new setting takes effect. Fix all of the following violations that were found against the API server:

- * a. Ensure the --authorization-mode argument includes RBAC
- * b. Ensure the --authorization-mode argument includes Node
- * c. Ensure that the --profiling argument is set to false

Fix all of the following violations that were found against the Kubelet:

- * a. Ensure the --anonymous-auth argument is set to false.
- * b. Ensure that the --authorization-mode argument is set to Webhook.

Fix all of the following violations that were found against the ETCD:

- * a. Ensure that the --auto-tls argument is not set to true

Hint: Take the use of Tool Kube-Bench

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

API server:

Ensure the --authorization-mode argument includes RBAC

Turn on Role Based Access Control. Role Based Access Control (RBAC) allows fine-grained control over the operations that different entities can perform on different objects in the cluster. It is recommended to use the RBAC authorization mode.

Fix - BuildtimeKubernetesapiVersion: v1

```
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    component: kube-apiserver
    tier: control-plane
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - command:
```

```
+ - kube-apiserver
+ - --authorization-mode=RBAC,Node
image: gcr.io/google_containers/kube-apiserver-amd64:v1.6.0
livenessProbe:
failureThreshold:8
httpGet:
host:127.0.0.1
path: /healthz
port:6443
scheme: HTTPS
initialDelaySeconds:15
timeoutSeconds:15
name: kube-apiserver-should-pass
resources:
requests: cpu: 250m
volumeMounts:
-mountPath: /etc/kubernetes/
name: k8s
readOnly:true
-mountPath: /etc/ssl/certs
name: certs
-mountPath: /etc/pki
name: pki
hostNetwork:true
volumes:
-hostPath:
path: /etc/kubernetes
name: k8s
-hostPath:
path: /etc/ssl/certs
name: certs
-hostPath:
path: /etc/pki
name: pki
```

Ensure the --authorization-mode argument includes Node

Remediation: Edit the API server pod specification file/etc/kubernetes/manifests/kube-apiserver.yaml on the master node and set the --authorization-mode parameter to a value that includeNs ode.

```
--authorization-mode=Node,RBAC
```

Audit:

```
/bin/ps -ef | grep kube-apiserver | grep -v grep
```

Expected result:

```
'Node,RBAC' has 'Node'
```

Ensure that the --profiling argument is set to false

Remediation: Edit the API server pod specification file/etc/kubernetes/manifests/kube-apiserver.yaml on the master node and set the below parameter.

```
--profiling=false
```

Audit:

```
/bin/ps -ef | grep kube-apiserver | grep -v grep
```

Expected result:

```
'false' is equal to 'false'
```

Fix all of the following violations that were found against the Kubelet:-

Ensure the --anonymous-auth argument is set to false.

Remediation: If using a Kubelet config file, edit the file to set authentication:anonymous: enabled to false. If using executable arguments, edit the kubelet service file

```
/etc/systemd/system/kubelet.service.d/10-kubeadm.conf
```

on each worker node and set the below parameter

```
in KUBELET_SYSTEM_PODS_ARGS
```

```
--anonymous-auth=false
```

variable.

Based on your system, restart the kubelet service. For example:

```
systemctl daemon-reload
```

```
systemctl restart kubelet.service
```

Audit:

```
/bin/ps -fC kubelet
```

Audit Config:

```
/bin/cat /var/lib/kubelet/config.yaml
```

Expected result:

```
'false' is equal to 'false'
```

*2) Ensure that the --authorization-mode argument is set to Webhook.

Audit

```
docker inspect kubelet | jq -e '[0].Args[] | match("--authorization-mode=Webhook").string'
```

Returned Value: --authorization-mode=Webhook

Fix all of the following violations that were found against the ETCD:

*a. Ensure that the --auto-tls argument is not set to true

Do not use self-signed certificates for TLS. etcd is a highly-available key value store used by Kubernetes deployments for persistent storage of all of its REST API objects. These objects are sensitive in nature and should not be available to unauthenticated clients. You should enable the client authentication via valid certificates to secure the access to the etcd service.

Fix - BuildtimeKubernetesapiVersion: v1

```
kind: Pod
```

```
metadata:
```

```
annotations:
```

```
scheduler.alpha.kubernetes.io/critical-pod:""
```

```
creationTimestamp: null
```

```
labels:
```

```

component: etcd
tier: control-plane
name: etcd
namespace: kube-system
spec:
containers:
-command:
+ - etcd
+ - --auto-tls=true
image: k8s.gcr.io/etcd-amd64:3.2.18
imagePullPolicy: IfNotPresent
livenessProbe:
exec:
command:
- /bin/sh
- -ec
- ETCDCTL_API=3 etcdctl --endpoints=https://[192.168.22.9]:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt
--cert=/etc/kubernetes/pki/etcd/healthcheck-client.crt --key=/etc/kubernetes/pki/etcd/healthcheck-client.key get foo
failureThreshold:8
initialDelaySeconds:15
timeoutSeconds:15
name: etcd-should-fail
resources: {}
volumeMounts:
-mountPath: /var/lib/etcd
name: etcd-data
-mountPath: /etc/kubernetes/pki/etcd
name: etcd-certs
hostNetwork:true
priorityClassName: system-cluster-critical
volumes:
-hostPath:
path: /var/lib/etcd
type: DirectoryOrCreate
name: etcd-data
-hostPath:
path: /etc/kubernetes/pki/etcd
type: DirectoryOrCreate
name: etcd-certs
status: {}

```

NEW QUESTION 3

Create a PSP that will prevent the creation of privileged pods in the namespace.
 Create a new PodSecurityPolicy named prevent-privileged-policy which prevents the creation of privileged pods.
 Create a new ServiceAccount named psp-sa in the namespace default.
 Create a new ClusterRole named prevent-role, which uses the newly created Pod Security Policy prevent-privileged-policy.
 Create a new ClusterRoleBinding named prevent-role-binding, which binds the created ClusterRole prevent-role to the created SA psp-sa.
 Also, Check the Configuration is working or not by trying to Create a Privileged pod, it should get failed.

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Create a PSP that will prevent the creation of privileged pods in the namespace.

```
$ cat clusterrole-use-privileged.yaml
```

```

--
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
name: use-privileged-psp
rules:
- apiGroups: ['policy']
resources: ['podsecuritypolicies']
verbs: ['use']
resourceNames:
- default-psp
--

```

```

apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
name: privileged-role-bind
namespace: psp-test
roleRef:
apiGroup: rbac.authorization.k8s.io
kind: ClusterRole
name: use-privileged-psp
subjects:
- kind: ServiceAccount
name: privileged-sa
$ kubectl -n psp-test apply -f clusterrole-use-privileged.yaml

```

After a few moments, the privileged Pod should be created.

Create a new PodSecurityPolicy named prevent-privileged-policy which prevents the creation of privileged pods.

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: example
spec:
  privileged: false # Don't allow privileged pods!
  # The rest fills in some required fields.
  seLinux:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  runAsUser:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
  volumes:
    - '*'
```

And create it with kubectl:

```
kubectl-admin create -f example-ppsp.yaml
```

Now, as the unprivileged user, try to create a simple pod:

```
kubectl-user create -f-<<EOF
```

```
apiVersion: v1
kind: Pod
metadata:
  name: pause
spec:
  containers:
  - name: pause
    image: k8s.gcr.io/pause
EOF
```

The output is similar to this:

```
Error from server (Forbidden): error when creating "STDIN": pods "pause" is forbidden: unable to validate against any pod security policy: []
```

Create a new ServiceAccount named psp-sa in the namespace default.

```
$ cat clusterrole-use-privileged.yaml
```

```
--
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: use-privileged-ppsp
rules:
- apiGroups: ['policy']
  resources: ['podsecuritypolicies']
  verbs: ['use']
  resourceNames:
  - default-ppsp
--
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: privileged-role-bind
  namespace: psp-test
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: use-privileged-ppsp
subjects:
- kind: ServiceAccount
  name: privileged-sa
```

```
$ kubectl -n psp-test apply -f clusterrole-use-privileged.yaml
```

After a few moments, the privileged Pod should be created.

Create a new ClusterRole named prevent-role, which uses the newly created Pod Security Policy prevent-privileged-policy.

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: example
spec:
  privileged: false # Don't allow privileged pods!
  # The rest fills in some required fields.
  seLinux:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  runAsUser:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
  volumes:
    - '*'
```

And create it with kubectl:

```
kubectl-admin create -f example-ppsp.yaml
```

Now, as the unprivileged user, try to create a simple pod:

```
kubectl-user create -f-<<EOF
apiVersion: v1
kind: Pod
metadata:
name: pause
spec:
containers:
- name: pause
image: k8s.gcr.io/pause EOF
The output is similar to this:
Error from server (Forbidden): error when creating "STDIN": pods "pause" is forbidden: unable to validate against any pod security policy: []
Create a new ClusterRoleBinding named prevent-role-binding, which binds the created ClusterRole prevent-role to the created SA psp-sa.
apiVersion:rbac.authorization.k8s.io/v1
# This role binding allows "jane" to read pods in the "default" namespace.
# You need to already have a Role named "pod-reader" in that namespace.
kind:RoleBinding
metadata:
name:read-pods
namespace:default
subjects:
# You can specify more than one "subject"
-kind:User
name:jane# "name" is case sensitive
apiGroup:rbac.authorization.k8s.io
roleRef:
# "roleRef" specifies the binding to a Role / ClusterRole
kind:Role#this must be Role or ClusterRole
name:pod-reader# this must match the name of the Role or ClusterRole you wish to bind to
apiGroup:rbac.authorization.k8s.io apiVersion:rbac.authorization.k8s.io/v1
kind:Role
metadata:
namespace:default
name:pod-reader
rules:
- apiGroups:[""]# "" indicates the core API group
resources:["pods"]
verbs:["get","watch","list"]
```

NEW QUESTION 4

Using the runtime detection tool Falco, Analyse the container behavior for at least 20 seconds, using filters that detect newly spawning and executing processes in a single container of Nginx.
store the incident file at /opt/falco-incident.txt, containing the detected incidents. one per line, in the format [timestamp],[uid],[processName]

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Send us your feedback on it.

NEW QUESTION 5

On the Cluster worker node, enforce the prepared AppArmor profile

```
#include<tunables/global>
profile nginx-deny flags=(attach_disconnected) {
#include<abstractions/base>
file,
# Deny all file writes.
deny/** w,
}
EOF'
```

Edit the prepared manifest file to include the AppArmor profile.

```
apiVersion: v1
kind: Pod
metadata:
name: apparmor-pod
spec:
containers:
- name: apparmor-pod
image: nginx
```

Finally, apply the manifests files and create the Pod specified on it. Verify: Try to make a file inside the directory which is restricted.

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Send us your Feedback on this.

NEW QUESTION 10

.....

Thank You for Trying Our Product

We offer two products:

1st - We have Practice Tests Software with Actual Exam Questions

2nd - Questions and Answers in PDF Format

CKS Practice Exam Features:

- * CKS Questions and Answers Updated Frequently
- * CKS Practice Questions Verified by Expert Senior Certified Staff
- * CKS Most Realistic Questions that Guarantee you a Pass on Your FirstTry
- * CKS Practice Test Questions in Multiple Choice Formats and Updatesfor 1 Year

100% Actual & Verified — Instant Download, Please Click
[Order The CKS Practice Test Here](#)